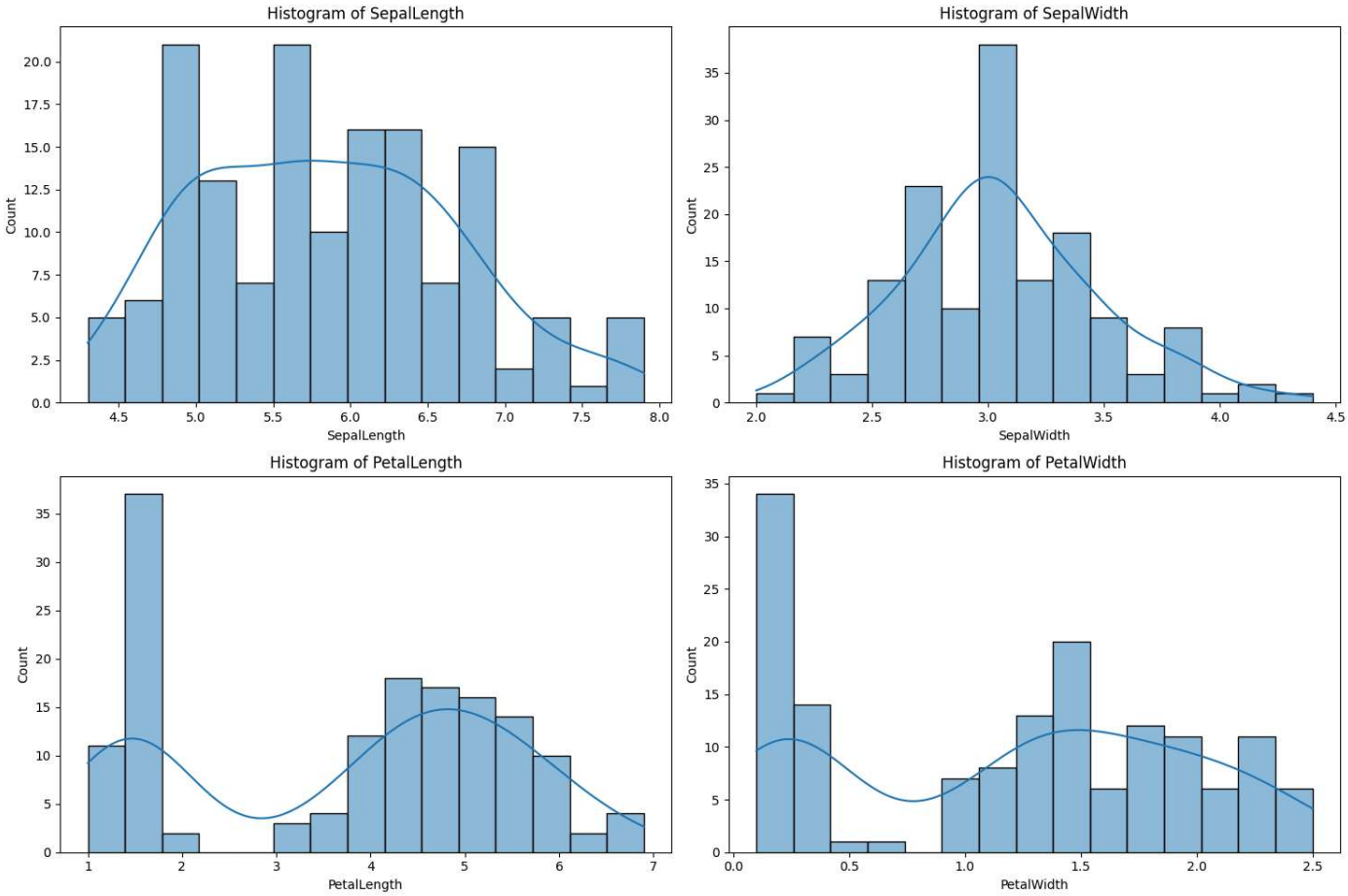
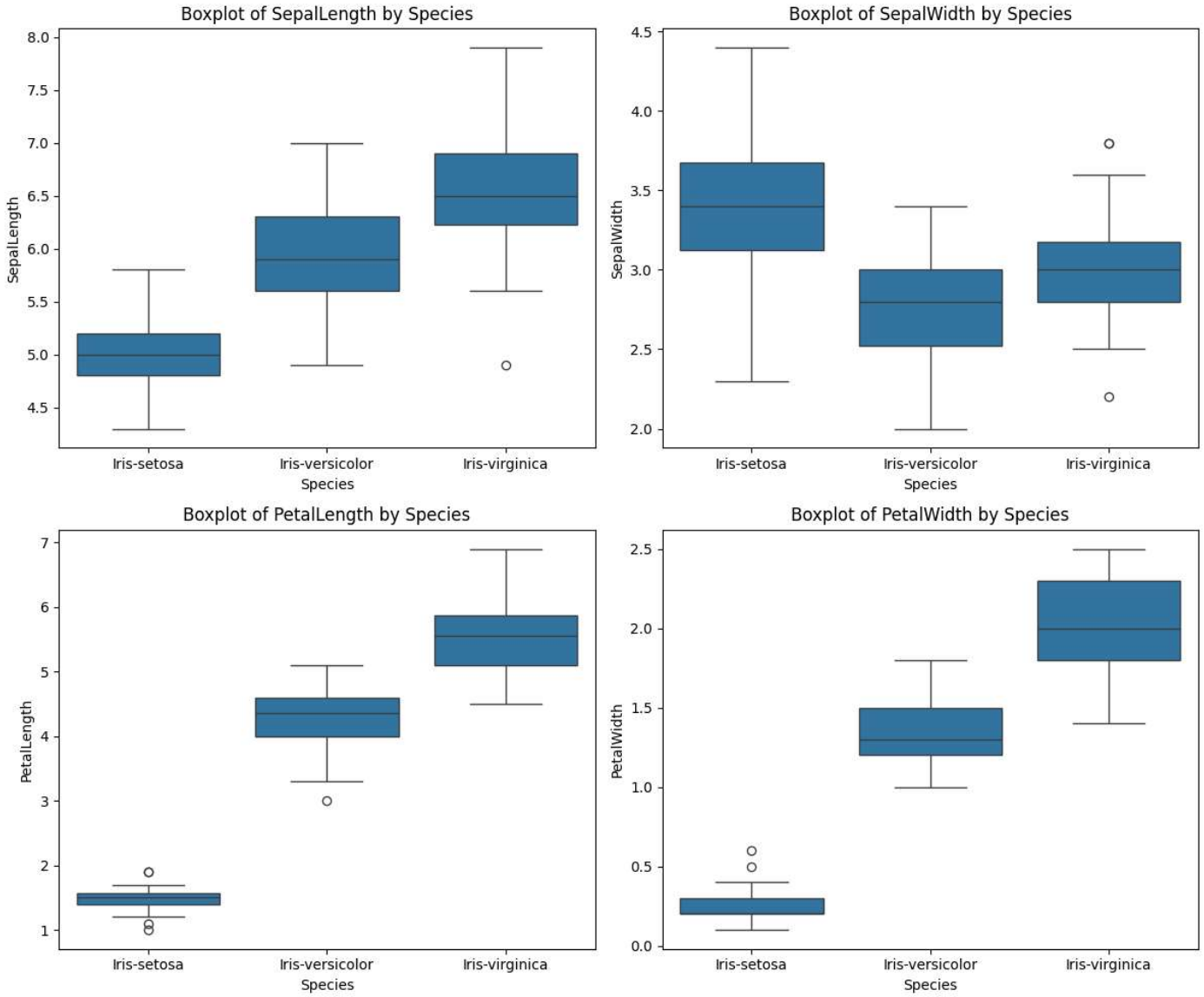
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.0.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.5
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->
```

```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
column_names = ['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species']
data = pd.read_csv(url, header=None, names=column_names)


features_types = {
    'SepalLength': 'numeric',
    'SepalWidth': 'numeric',
    'PetalLength': 'numeric',
    'PetalWidth': 'numeric',
    'Species': 'nominal'  # Categorical type
}


data_features = ['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth']



plt.figure(figsize=(15, 10))
for i, feature in enumerate(data_features, 1):
    plt.subplot(2, 2, i)
    sns.histplot(data[feature], kde=True, bins=15)
    plt.title(f"Histogram of {feature}")
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(12, 10))
for i, feature in enumerate(data_features, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(x='Species', y=feature, data=data)
    plt.title(f"Boxplot of {feature} by Species")

plt.tight_layout()
plt.show()
```




```
outliers = {}
for feature in data_features:
    outliers[feature] = {
        "Q1": np.percentile(data[feature], 25),
        "Q3": np.percentile(data[feature], 75),
        "IQR": np.percentile(data[feature], 75) - np.percentile(data[feature], 25),
        "Lower Bound": np.percentile(data[feature], 25) - 1.5 * (np.percentile(data[feature], 75) - np.percentile(data[feature], 25)),
        "Upper Bound": np.percentile(data[feature], 75) + 1.5 * (np.percentile(data[feature], 75) - np.percentile(data[feature], 25)),
    }

print(outliers)
```

```
{'SepalLength': {'Q1': np.float64(5.1), 'Q3': np.float64(6.4), 'IQR': np.float64(1.3000000000000007), 'Lower Bound': np.float64(3.149999
```

```
def identify_outliers(data, features):
    outliers = {}
    for feature in features:
        # Calculate Q1 (25th percentile), Q3 (75th percentile), and IQR
        Q1 = np.percentile(data[feature], 25)
        Q3 = np.percentile(data[feature], 75)
```

```
        IQR = Q3 - Q1

        # Calculate the lower and upper bounds
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Identify outliers
        outlier_data = data[(data[feature] < lower_bound) | (data[feature] > upper_bound)]
        outliers[feature] = outlier_data

    return outliers

# Identify outliers in the dataset
outliers = identify_outliers(data, data_features)

# Print outliers for each feature
for feature, outlier_data in outliers.items():
    print(f"Outliers in {feature}:\n", outlier_data, "\n")
```

```
Outliers in SepalLength:
 Empty DataFrame
Columns: [SepalLength, SepalWidth, PetalLength, PetalWidth, Species]
Index: []

Outliers in SepalWidth:
     SepalLength  SepalWidth  PetalLength  PetalWidth          Species
15           5.7         4.4          1.5         0.4      Iris-setosa
32           5.2         4.1          1.5         0.1      Iris-setosa
33           5.5         4.2          1.4         0.2      Iris-setosa
60           5.0         2.0          3.5         1.0  Iris-versicolor

Outliers in PetalLength:
 Empty DataFrame
Columns: [SepalLength, SepalWidth, PetalLength, PetalWidth, Species]
Index: []

Outliers in PetalWidth:
 Empty DataFrame
Columns: [SepalLength, SepalWidth, PetalLength, PetalWidth, Species]
Index: []
```

```
sns.pairplot(data, hue='Species')
plt.suptitle('Pairplot of Numerical Features')
plt.show()
```